Why Should Anyone use Colours?

or, Syntax Highlighting Beyond Code Snippets

Marco Patrignani[†,‡]
[†] *Stanford University*
[‡] *CISPA Helmholtz Centre for Information Security*
mp@cs.stanford.edu

Abstract

Syntax highlighting in the form of colours and font diversification, is an excellent tool to provide clarity, concision and correctness to writings. Unfortunately, this practice is not widely adopted, which results in often hard-to-parse papers. The reasons for this lack of adoption is that researchers often struggle to embrace new technologies, piling up unconvincing motivations.

This paper argues against such motivations and justifies the usage of syntax highlighting so that it can become a new standard for dissemination of clearer and more understandable research. Moreover, this paper reports on the criticism grounded on the shortcomings of using syntax highlighting in LaTeX and suggests remedies to that.

We believe this paper can be used as a guide to using syntax highlighting as well as a reference to counter unconvincing motivations against it.

1 Introduction

Syntax highlighting (SH), in the form of colours and font diversification, is an excellent tool to provide clarity, concision and correctness to writings. Generally, in most webpages hypertext links are coloured in blue or underlined (whereas normal text is black and not underlined). That syntax highlighting is done for the sake of clarity: writers put effort in devising links that readers can follow to further their knowledge. More specifically, many researchers are already used to seeing a form of syntax highlighting, for example, in code snippets. There, keywords, identifiers, literal and comments are all given different colours and different fonts, as the familiar code snippet below points out.

```
1 public static void main (int* args) { return 0; } // comment
```

However, this is not done widely, and some areas of research suffer from this lack of practice. Let us now consider and example from computer science literature, specifically, when describing a compiler from a language L_s to another language L_t.[1]

[1] Several other examples exist, for example comparing different systems, or models, or languages for expressive power, one often has to indeed first define both systems. We treat those examples later in the paper.

> *1) A compiler $[\![\cdot]\!]$ is a function from programs P of language S to programs P of language T, which we denote as $[\![P]\!]$.*
>
> *2) The behaviour of a program P is a set of traces defined as $\mathtt{B}(P)$.*
>
> *3) For a correct compiler, the behaviour of a compiled program $[\![P]\!]$ is a subset of the behaviour of the original program P, i.e., $\mathtt{B}([\![P]\!]) \subseteq \mathtt{B}(P)$.*

This example already shows some shortcomings of not using any form of syntax highlighting. In point 1, the same meta-variable P denotes both elements of a source language L_s and elements of a target language L_t, which can be very different between each other. A simple solution is that to annotate said meta-variable P with a subscript S or T to attribute it to the right language. However, this has several drawbacks, as it often ends up polluting equations and formulas with additional subscripts.

A much more compact solution would be to just write all elements of S and of T in two different colours and fonts. For example, we can write all elements of S in a blue, sans-serif font and all elements of T in a **red**, **bold** one.[2] The result is much more concise and clear.

> *1) A compiler $[\![\cdot]\!]$ is a function from programs P of language S to programs $\mathbf{P}$ of language $\mathbf{T}$, which we denote as $[\![\mathsf{P}]\!]$.*

Another crafty shortcoming of not using any form of SH can be seen in point 3. There, the equation $\mathtt{B}([\![P]\!]) \subseteq \mathtt{B}(P)$ is seemingly correct. However, if we try to colour it, we realise something is afoot: $\mathbf{B}([\![\mathsf{P}]\!]) \subseteq \mathsf{B}(\mathsf{P})$, we are comparing sets whose elements are *different*.

Of course without any SH this is rather tricky to notice, while with basic highlighting such as colours it is not. Again, some people may advocate resorting to subscripts, but this complicates things. In fact one first has to generalise point 2 and the notation for behaviours as follows.

> *2-b) The behaviour of a program P_S of language S is a set of traces defined as $\mathtt{B}_\mathtt{S}(P_S)$.*

Then, one can write point 3 with subscripts, as follows:

$$\mathtt{B}_\mathtt{T}([\![P_S]\!]) \subseteq \mathtt{B}_\mathtt{S}(P_S)$$

Again, this solution is not concise and it does not scale if the formula at hand has several elements from the different languages. This can be seen in the next example that compares behaviours of two programs linked via operator + (which can vary wildly between S and $\mathbf{T}$).[3] For clarity we report the syntax-highlighted version too.

$$\mathtt{B}_\mathtt{T}\left([\![P^1_S]\!] +_T [\![P^2_S]\!]\right) \subseteq \mathtt{B}_\mathtt{S}\left(P^1_S +_S P^2_S\right)$$

[2] The choice of this specific syntax highlighting scheme is motivated later in this paper.

[3] This example also highlights that subscripts "consume" another common place for identifiers, if we had a matrix of programs P^j_i, where to put the language annotation would not be obvious.

$$\mathbf{B}\,(\llbracket \mathsf{P}_1 \rrbracket + \llbracket \mathsf{P}_2 \rrbracket) \subseteq \mathsf{B}\,(\mathsf{P}_1 + \mathsf{P}_2)$$

Unfortunately, the usage of SH is not widespread and often researchers struggle to embrace these technologies, piling up unconvincing motivations. The most notorious ones are that colours are obscure to many reviewers (and we all know that baffling reviewers is to be avoided), that they will be hard to discern in printouts and that colourblind people will be hindered by them.

Contributions & Outline This paper meticulously argues against such motivations and justifies the usage of SH (in the form of colours and font diversification) so that it can become a new standard for dissemination of clearer and more understandable research.

To motivate the use of SH, Section 2 argues in more depth about how to use colours and font diversification to improve clarity, concision and correctness. Then, Section 3 presents often-voiced criticism about the usage of SH and argues against it. Finally, Section 4 discusses the main shortcomings of using SH in the form of colours and font diversification with LaTeX and describes solutions to said shortcomings.

2 Why Bothering

Throughout this section we will provide examples of when to use SH in the form of colours and font differentiation to increase clarity, conciseness and correctness of papers. The following examples (Examples 1 to 3) discuss respectively the case of extending a system, of identifying different parts of a system and of identifying different systems.

Example 1 (Additions to a system). When presenting an enrichment to a system, be it in terms of a program logic, a type system or a new semantics, it is often useful to highlight said enrichment to the reader.

In the case of program logics, this has the distinct benefit of clearly demarcating what must be supplied by possibly different entities, namely the program and the annotations.

$$\frac{\{\mathsf{p}\}\ c\ \{\mathsf{q}\}}{\{\mathsf{p} * \mathsf{r}\}\ c\ \{\mathsf{q} * \mathsf{r}\}}\ (\text{Frame}) \qquad \frac{}{\{\ell \mapsto \mathsf{u}\}\ \ell := v\ \{\ell \mapsto \mathsf{v}\}}\ (\text{Assignment})$$

In the case of type systems, this has already been pointed out to immediately highlight the Curry-Howard isomorphism (Wadler, 2007). In fact, by erasing the appropriate parts of the type-system, a logic appears (Bernardy and Guilhem, 2013), and one can see the erasure simply by putting on glasses of the same colour as the syntax (blue in the case of the snippet below).

$$\frac{\mathsf{f} : A \to B \qquad \mathsf{v} : A}{\mathsf{f}\ \mathsf{v} : B}\ (\text{Type-Lam}) \qquad \frac{A \to B \qquad A}{B}\ (\text{Modus Ponens})$$

Finally, consider a language and its operational semantics (indicated as $\longrightarrow$) being extended with another semantics with labels (indicated as $\xrightarrow{1}$), that relies on the former. In order to highlight where these dependencies happen we may want to colour semantics rules of the latter where this dependency is.

$$\frac{e \longrightarrow v}{e \xrightarrow{1} v}\ \text{(Connecting Step)}$$

Moreover, we may want to state something about the semantics, e.g., that a term reducing according to the first also reduces according to the latter.

$$t \longrightarrow v \Rightarrow t \longrightarrow v$$

In this case SH helps ensuring that the reader knows that it is the left hand side of the equation that is concerned with the latter semantics. Crucially, this is achieved without polluting the notation with unpleasant and confusing superscripts and subscripts, which may be used to denote other useful information (e.g., number of steps). ⊡

Example 2 (Different parts of a system). Another use case for SH comes from the world of compilation, as presented in Section 1 (Patrignani et al., 2016). Consider a modular compiler, i.e., one that operates on components and then links together its output to produce a larger program. If we want to reason about its correctness we may want to express something like the following:

$$\mathbf{B}\,(\llbracket \mathsf{C}_1 \rrbracket + \cdots + \llbracket \mathsf{C}_n \rrbracket) \subseteq \mathsf{B}\,(\mathsf{C}_1 + \cdots + \mathsf{C}_n)$$

Here, colours provide a crucial highlight, namely that there is a part of the two systems, which we write as $+$ that is not the same between the two systems. In fact, $+$ may be the indication of a linker, and the usage of SH clarifies that there are *different* kinds of linkers being employed in the two languages.

Abstracting away from details of the system at hand, we may be using the same notation to reason about the meta-system, e.g., to reason about equality of programs of the system. This is, for example, the case for the statement of fully abstract compilation (Abadi, 1998):[4]

$$\forall \mathsf{P}_1, \mathsf{P}_2.\, \mathsf{P}_1 \simeq \mathsf{P}_2 \iff \llbracket \mathsf{P}_1 \rrbracket \simeq \llbracket \mathsf{P}_2 \rrbracket$$

In this case, the relation $\simeq$ that we use to relate programs can vary between the two languages, and the SH points that out nicely to the reader. ⊡

Example 3 (Different systems interacting with each other). Finally, we may be interested in capturing how different systems interact with each other, for example to study the foreign-function interface (FFI) of said systems. This

[4] Fortunately, many articles on secure compilation (of which fully abstract compilation is an instance) use syntax highlighting (Abate et al., 2019; New et al., 2016; Patrignani and Garg, 2019; Patrignani et al., 2016; Patterson et al., 2017; Scherer et al., 2018).

is often called a multilanguage system and it is indeed a scenario where colours have successfully been used to achieve clarity (Matthews and Findler, 2007; Patterson et al., 2017; Perconti and Ahmed, 2014). In a multilanguage system where languages I and **F** are calling each other, the FFI from the former to the latter is written as IF and the reverse is written as FI. Thus, one can write snippets such as the following one, where some imperative code calls a **functional snippet**.

$$\mathsf{unsafe(x)\{if\ x\ then}\ \ (IF(\boldsymbol{\lambda y.\, y{==}2})\,(FI\mathsf{x}))\ \ \mathsf{else\ skip\}}$$

The explicit boundaries as well as the SH helps the reader understanding why suddenly there is some functional code in the middle of imperative one. ⊡

3 Criticism to Syntax Highlighting

This section dissects the most often-heard criticism to the adoption of SH, i.e., that colourblind people are hindered by it (Section 3.1) and black and white printers do not render SH nicely (Section 3.2). This section also discusses the hardest criticism coming from people that simply are not used to SH (Section 3.3).

3.1 Colourblind

Colourblindness is a serious genetic condition that affects the 8% of male population worldwide and a smaller fraction of female population, as the gene responsible for colourblindness is found in the X chromosome. Thus, there is a significant chance that colourblind people will attend your talks and read your papers and this is what many researchers point out to people using colours. Sadly, this is also where most researchers stop, not noticing that effective measures can be taken to ensure that colourblind people also enjoy coloured papers. To ensure colourblind people can make out what is going on when colours are used, two things can be done:

1. use font diversification;
2. use colours that are still distinguishable for colourblind people.

Concerning point 1, that is why throughout this paper we use sans-serif and **bold** fonts. Concerning point 2, there exist tools such as Sim Daltonism[5] that allows us to conduct an analysis on what colours can still be perceived by people with different kinds of colourblindness. The combination blue versus **red** is, for example, still discernible from *all* kind of colourblind people (who can still perceive colours). So it is clear: using a blue, sans-serif font and a **red**, **bold** one voids any claim that the paper is inaccessible to colourblind people.

A more advanced concern pops up now: what if there are more than two systems in our paper? What is the best combination for a third system, and a

[5] Available for MAC OS X at: https://michelf.ca/projects/sim-daltonism/

fourth one and so on? Unfortunately, there are only so many colour and font combination available. Luckily writing papers with more than two systems is already challenging, more than three is very rare and more than four is seemingly unheard of. The list below summarises SH per number of systems, while the table below shows the SH in action on different kinds of text: characters, words, greek letters and math. The colour between brackets is the precise LaTeXcolour that each line uses.

1. blue (RoyalBlue), sans-serif;
2. **red** (**RedOrange**), **bold**, **with-serif**;
3. *emerald* (*Emerald*), *italic*, *with-serif*;
4. `orange` (`YellowOrange`), `monospaced`, `with-serif`;
5. pink (CarnationPink), underlined, sans-serif.

	First	Second	Third	Fourth	Fifth
Character	a z g	**a z g**	*a z g*	a z g	a z g
Word	gazebo	**gazebo**	*gazebo*	gazebo	gazebo
Greek	$\alpha\ \Pi$	$\boldsymbol{\alpha\ \Pi}$	$\alpha\ \mathit{\Pi}$	$\alpha\ \Pi$	$\alpha\ \Pi$
Symbol	$+\ \simeq$	$\boldsymbol{+\ \simeq}$	$+\ \simeq$	$+\ \simeq$	$+\ \simeq$

In order to understand how colourblind people perceive the use of SH, we provide the list of images below. Those images are obtained through the use of Sim-Daltonism, an application that replicates different forms of colourblindness. The images show how people with different visual impairment (in the title of the frames) see the table above. This list serves to inform readers without visual impairment that the choice of colours and of fonts still result in visually distinct syntax highlighting that colourblind people can benefit from.

Deuteranopia

	First	Second	Third	Fourth	Fifth
Character	a z g	**a z g**	*a z g*	a z g	a z g
Word	gazebo	**gazebo**	*gazebo*	gazebo	gazebo
Greek	$\alpha\ \Pi$	$\boldsymbol{\alpha\ \Pi}$	$\alpha\ \mathit{\Pi}$	$\alpha\ \Pi$	$\alpha\ \Pi$
Symbol	$+\ \simeq$	$\boldsymbol{+\ \simeq}$	$+\ \simeq$	$+\ \simeq$	$+\ \simeq$

Deuteranomaly

	First	Second	Third	Fourth	Fifth
Character	a z g	**a z g**	*a z g*	a z g	a z g
Word	gazebo	**gazebo**	*gazebo*	gazebo	gazebo
Greek	$\alpha\ \Pi$	$\boldsymbol{\alpha\ \Pi}$	$\alpha\ \mathit{\Pi}$	$\alpha\ \Pi$	$\alpha\ \Pi$
Symbol	$+\ \simeq$	$\boldsymbol{+\ \simeq}$	$+\ \simeq$	$+\ \simeq$	$+\ \simeq$

Protanopia

	First	Second	Third	Fourth	Fifth
Character	a z g	a z g	a z g	a z g	a z g
Word	gazebo	gazebo	gazebo	gazebo	gazebo
Greek	α Π	α Π	α Π	α Π	α Π
Symbol	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$

Protanomaly

	First	Second	Third	Fourth	Fifth
Character	a z g	a z g	a z g	a z g	a z g
Word	gazebo	gazebo	gazebo	gazebo	gazebo
Greek	α Π	α Π	α Π	α Π	α Π
Symbol	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$

Tritanopia

	First	Second	Third	Fourth	Fifth
Character	a z g	a z g	a z g	a z g	a z g
Word	gazebo	gazebo	gazebo	gazebo	gazebo
Greek	α Π	α Π	α Π	α Π	α Π
Symbol	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$

Tritanomaly

	First	Second	Third	Fourth	Fifth
Character	a z g	a z g	a z g	a z g	a z g
Word	gazebo	gazebo	gazebo	gazebo	gazebo
Greek	α Π	α Π	α Π	α Π	α Π
Symbol	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$

Monochromacy

	First	Second	Third	Fourth	Fifth
Character	a z g	a z g	a z g	a z g	a z g
Word	gazebo	gazebo	gazebo	gazebo	gazebo
Greek	α Π	α Π	α Π	α Π	α Π
Symbol	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$

Partial Monochromacy

	First	Second	Third	Fourth	Fifth
Character	a z g	a z g	a z g	a z g	a z g
Word	gazebo	gazebo	gazebo	gazebo	gazebo
Greek	α Π	α Π	α Π	α Π	α Π
Symbol	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$	$+$ $\simeq$

3.2 Black and White Printouts

Some researchers argue that when printing on black and white settings, there is not much difference between the shades of grey to effectively distinguish the colours. This seems a rather short-lived comment for several reasons.

First, font diversification carries out to black and white printouts, so already the font gives meaning to distinguish elements with similar shades of grey. This is particularly visible in the second-to-last picture from the list above — labelled Monochromacy — which replicates how people that can only distinguish shades of grey see the syntax highlighting we propose. Such a colouring is exactly what a black and white printer also produces.

Second, and most importantly, paper printouts are a dying medium and we should encourage its death. With the advent of modern technologies such as hi-resolution tablets, it seems an environmentally unhealthy choice to keep printing papers that will be read once, and then recycled (which is still a costly, albeit noble, process).

3.3 Accepting a New Notation

The hardest criticism to using and employing colours is, often, simply that of a mental barrier. This is still a rather new notation, with a slowly-growing and still small group of adopters. Although it takes a little effort to get into the colour-usage mentality, many are simply unwilling to take this step.

The best way to convert these researchers is to adopt this notation ourselves, use it and make it mainstream, showing clear, concise, elegant work. Then they will realise it takes a little step to understand (and use) colours, and they have already done that step by looking at our work that does already use colour.

4 Colours and LaTeX

Using colours to typeset papers in LaTeX can be rather complex and lead to numerous concerns. Fortunately, this section presents some notorious issues when compiling coloured code and solutions to them.

4.1 Colour Choice

A first concern is that the standard LaTeX colours are not very pleasant: they are very bright, almost too vibrant colours and a fully-coloured page would hurt the eyes. The best solution to this is to use the `xcolor` package, which provides RoyalBlue replacing the latex blue, **RedOrange** replacing latex red and Forest Green replacing latex green among many other more pleasant hues such as Carnation Pink for pink text.

So, do not forget to include:

```
\usepackage[usenames,dvipsnames]{xcolor}
```

4.2 Colouring Macros

Once the `xcolor` package is loaded, we can define commands that perform syntax highlighting. To keep in line with the compilation examples from Section 1, we define two commands for typesetting blue source and red target elements:

```
\newcommand{\src}[1]{\color{RoyalBlue}{\mathsf{#1}}}
\newcommand{\trg}[1]{\color{RedOrange}{\mathbf{#1}}}
```

A pre-made structuring of macros that perform syntax highlighting can be found in the sources of this paper on arXiv or at the following link: http://theory.stanford.edu/~mp/mp/Publications_files/cmds.tex.

4.3 Colours in Titles and Headings

Sometimes having a `\src{.}` (or an analogous SH command) inside a section title will make LaTeX not build. In that case one has to redefine the colour name. For example, for a colour called *RoyalBlue* one has to write the following to ensure LaTeXwill build.

```
\colorlet{ROYALBLUE}{RoyalBlue}
```

4.4 Math, Symbols, Alignments and Colours

It is indeed complicated to ensure macros do not break when used within equations and aligned environments in LaTeX. Here are few tips to ensure one can write modular macros that will build in most settings.

1. Define symbols with their colouring.

   For example, if you need to define a relation such as $\approx$ (`\approx`) in different colours, it will come in handy to have $\srcapprox$ (`\srcapprox`) as a single new command.

2. Define bold symbols.

   Bold math is very complex to get right and robust, since it relies on the brittle `bm` package. As such, it is recommended to have a command for typesetting bold symbols in the second system:

   ```
   \newcommand{\trgbold}[1]{\color{RedOrange}{\mathbf{#\{1}}}}
   ```

   This way, when we need to write just text, e.g., **cat** we can write `\trg{cat}`, but when we are writing a symbol as $\trgbold{\approx}$, we instead write `\trgbold{\approx}`.

   (a) Nesting commands.

   In case you want to be able to nest commands where one may use `\bm`, other non-bold commands should contain `\unboldmath`, as follows:

   ```
   \newcommand{\srcu}[1]{\color{RoyalBlue}{\ensuremath{\unboldmath
   \({\mathsf{#1}}\)}}}
   ```

   This way you can carelessly write a top-level `\trgbold` and supply `\srcu` text as part of the argument. For example $\trgbold{\srcu{A} \approx \srcu{B}}$ is obtained by just writing `\trgbold{\srcu{A} \approx \srcu{B}}`

To ensure no LaTeX errors pop up, be sure to include this in your preamble *before* loading the `bm` package:[6]

```
\newcommand\hmmax0
\newcommand\bmmax0
```

3. Define black symbols.

   The same idea above applies to black symbols, e.g., having $[\![\cdot]\!]$ as a single new command. If we have a command `\blk` to write black text, we can define $[\![\cdot]\!]$ (comp) as follows:

   ```
   \newcommand{\blk}[1]{\color{black}{#1}}
   \newcommand{\comp}[1]{\blk{\left\llbracket\src{#1}\right\rrbracket}}
   ```

   That way one does not have to worry when embedding such a symbol in a larger coloured text such as $[\![C]\!]+C$, which we can simply write as:

   ```
   \trg{\comp{C} \trgbold{+} C}
   ```

4. No wrapping of parameters in SH.

   Let us assume we have a macro to typeset a let-in construct that takes the three parameters of the letin and produces the following: *let* $x = v$ *in* e. We may define such a macro as follows:

   ```
   \newcommand{\letin}[3]{let~#1=#2~in~#3}
   ```

   In order for it to appear correctly coloured, we have to provide a coloured variant as follows:

   ```
   \newcommand{\srcletin}[3]{\src{let}~#1\src{=}#2~\src{in}~#3}
   ```

   It is important not to wrap the arguments in `\src{·}` otherwise when used within an aligned environment, things may break. In fact, if we were to write

   ```
   \newcommand{\wrongsrcletin}[3]{\src{let~#1}\src{=#2}~\src{in~#3}}
   ```

   where the `\src{·}` commands encompasses the arguments too, those arguments will now by part of a sub-group. Thus we would not be able to place alignment markers such as "&" inside them, as in the example below. There, the third argument is "`\\ &\ x`" and this ensures a proper alignment of subparts of the command (as can be seen below).

   let x=v in
   x

```
\begin{align*}
&
\srcletin{\src{x}}{\src{v}}{
  \\
  &\
  \src{x}
}
\end{align*}
```

[6] Cfr: https://tex.stackexchange.com/questions/3676/too-many-math-alphabets-error

Unfortunately, not all that glitters is gold, which means we must wrap the contents of the arguments that are not alignment markers in `\src{·}`.

5 Conclusion

This paper argued in favour of using syntax highlighting in the form of colours and font diversification to provide more clarity, concision and correctness to research papers. This paper discussed the pros of using colours and font diversification, how to prevent poorly-grounded criticism against it and how to overcome technical difficulties when typesetting coloured work with LaTeX.

Acknowledgement: This work has benefitted from discussions with the following people as well as from collaborating with a subset of them (and from the many reviewer comments we had for said collaborations): Carmine Abate, Amal Ahmed (my first reference for colour usage), Roberto Blanco, William Bowman, Dave Clarke, Stefan Ciobaca (thanks for pointing out the usage of `\unboldmath`) Dominique Devriese, Deepak Garg, Catalin Hritcu, Adriaan Larmuseau, Max New, Daniel Patterson, Frank Piessens, Jeremy Thibault.

References

Abadi, Martín (1998). "Protection in Programming-Language Translations". In: *ICALP'98*, pp. 868–883.

Abate, Carmine, Roberto Blanco, Deepak Garg, Cătălin Hriţcu, Marco Patrignani, and Jérémy Thibault (2019). "Journey Beyond Full Abstraction: Exploring Robust Property Preservation for Secure Compilation". In: *2019 IEEE 32th Computer Security Foundations Symposium*. CSF 2019.

Bernardy, Jean-Philippe and Moulin Guilhem (2013). "Type-theory in Color". In: *SIGPLAN Not.* 48.9, pp. 61–72. ISSN: 0362-1340. DOI: 10.1145/2544174.2500577. URL: http://doi.acm.org/10.1145/2544174.2500577.

Matthews, Jacob and Robert Bruce Findler (2007). "Operational Semantics for Multi-language Programs". In: *SIGPLAN Not.* 42.1, pp. 3–10. ISSN: 0362-1340. DOI: 10.1145/1190215.1190220. URL: http://doi.acm.org/10.1145/1190215.1190220.

New, Max S., William J. Bowman, and Amal Ahmed (2016). "Fully Abstract Compilation Via Universal Embedding". In: *21st ACM SIGPLAN International Conference on Functional Programming, ICFP*, pp. 103–116. DOI: 10.1145/2951913.2951941. URL: https://www.williamjbowman.com/resources/fabcc-paper.pdf.

Patrignani, Marco and Deepak Garg (2019). "Robustly Safe Compilation". In: *Programming Languages and Systems - 28th European Symposium on Programming, ESOP 2019*. ESOP'19. URL: https://arxiv.org/abs/1804.00489.

Patrignani, Marco, Dominique Devriese, and Frank Piessens (2016). "On Modular and Fully Abstract Compilation". In: *Computer Security Foundations Symposium*.

Patterson, Daniel, Jamie Perconti, Christos Dimoulas, and Amal Ahmed (2017). "FunTAL: Reasonably Mixing a Functional Language with Assembly". In: *SIGPLAN Not.* 52.6, pp. 495–509. ISSN: 0362-1340. DOI: 10.1145/3140587.3062347. URL: http://doi.acm.org/10.1145/3140587.3062347.

Perconti, James T. and Amal Ahmed (2014). "Verifying an Open Compiler Using Multi-language Semantics". In: *Proceedings of the 23rd European Symposium on Programming Languages and Systems - Volume 8410*. New York, NY, USA: Springer-Verlag New York, Inc., pp. 128–148. ISBN: 978-3-642-54832-1. DOI: 10.1007/978-3-642-54833-8_8. URL: http://dx.doi.org/10.1007/978-3-642-54833-8_8.

Scherer, Gabriel, Max New, Nick Rioux, and Amal Ahmed (2018). "Fabous Interoperability for ML and a Linear Language". In: *Foundations of Software Science and Computation Structures*. Ed. by Christel Baier and Ugo Dal Lago. Cham: Springer International Publishing, pp. 146–162. ISBN: 978-3-319-89366-2.

Wadler, Philip (2007). "The Girard—Reynolds Isomorphism (Second Edition)". In: *Theor. Comput. Sci.* 375.1-3, pp. 201–226. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2006.12.042. URL: http://dx.doi.org/10.1016/j.tcs.2006.12.042.